实验四　锁存器和触发器

2020 年秋季学期

杨飞洋 191220138

目录

1. 实验目的：

复习锁存器和触发器的工作原理，学习 verilog 语言对于时序电路的描述。学习如何对时序电路进行仿真，了解 Verilog 语言中阻塞赋值语句和非阻塞赋值语句的区别。比较同步清零和异步清零的区别，学习 Verilog HDL 模块实例化的方法。

2. 实验原理（知识背景）

锁存器和触发器是时序电路的基本构件。锁存器和触发器都是由独立的逻辑门电路和反馈电路构成的，锁存器在时钟信号为有效电平的整个时间段，不断监测其所有的输入端，此段时间内的任何满足输出改变条件的输入，都会改变输出；触发器只有在时钟信号变化的瞬间才改变输出值。

3. 实验环境/器材

系统：windows10
开发软件：Quartus 17.1 Lite
开发板：DE10 Standard
仿真环境：ModelSim
芯片：Cyclone V ， 5CSXFC6D6

4.设计思路

第一部分：
赋值符号“=”：阻塞赋值语句
赋值符号“<=”：非阻塞赋值语句
写两个功能相似的模板，只是在赋值符号上有所不同，再在 RTL viewer 中查看区别即可。
综合两个级联的触发器,输出为 out_qt1 的触发器的输入为 in_d，而输出 out_qt2 的触发器输入信号是 out_qt1。
模块代码：

```
module d_trigger(clk,en,in_d,out_qt1,out_qt2);
   input clk;
   input en;
   input in_d;
   output reg out_qt1;
   output reg out_qt2;

   always @(posedge clk)
      if(en) begin
         out_qt1 = in_d;
         out_qt2 = out_qt1;
         end
      else begin
         out_qt1 = out_qt1;
         out_qt2 = out_qt2;
         end

endmodule
```

```
module d_trigger(clk,en,in_d,out_qt1,out_qt2);
   input clk;
   input en;
   input in_d;
   output reg out_qt1;
   output reg out_qt2;

   always @(posedge clk)
      if(en) begin
         out_qt1 <= in_d;
         out_qt2 <= out_qt1;
         end
      else begin
         out_qt1 <= out_qt1;
         out_qt2 <= out_qt2;
         end

endmodule
```

再在顶层文件中调用模块，即可。

第二部分：
异步清零可以不顾时钟信号，只要清零信号到来就进行清零操作。同步清零即使清零信号有效也要等时钟信号有效沿到来时才清零。所以代码的唯一区别就是赋值条件的区别。
即 always @(posedge clk or posedge clr) 和 always @(posedge clk)的区别。
下图是模块的代码。

异步：

```
module a_reset(clk,clr,in_d,out_t);
   input clk;
   input clr;
   input in_d;
   output reg out_t;

   always @(posedge clk or posedge clr)
      if(clr)
         out_t <= 0;
      else
         out_t <= in_d;

endmodule
```

同步：

```
module s_reset(clk,clr,in_d,out_t);
   input clk;
   input clr;
   input in_d;
   output reg out_t;

   always @(posedge clk)
      if(clr)
         out_t <= 0;
      else
         out_t <= in_d;

endmodule
```

5.实验步骤 （第一部分和第二部分类似）

一．利用 systembuilder 建立工程

选择 CLOCK、SWITCH、LEDR

二．编写 Verilog 代码

写一下相应模块的代码，在顶层文件中做相应的调用，进行编译调试，直至成功。

三．编写 test bench

利用 processing 中的 start test bench template writer 让系统自动生成 test bench 测试文件
在 simulation 中的 modelsim 文件中打开.vt 文件
修改仿真时间单位，`timescale 10 ns/ 1 ps
在 initial 下方 begin 和 end 之间写入测试数据
在 always 语句块中写入 #5 clk = ~clk;。

四．设置 test bench

在 assignments->settings->simulation->compile test bench 中设置 test bench，new 一个 test bench，注意将顶层实体修改为 filename_vlg_tst，
再将 simulation->modelsim 中的.vt 文件 add 进去

五．仿真测试

利用 RTL Simulation 调用 modelsim
在波形图中 zoom full 就可以看到测试结果

6.测试方法

切换状态，第一部分注意将阻塞和非阻塞的区别显示出来，第二部分注意将同步清零和异步清零的区别显示出来。

第一部分（阻塞和非阻塞），SW[0]为使能端，SW[1]为输入端

```
begin
// code that executes only once
// insert code here --> begin
CLOCK_50 = 0;SW[1] = 0;SW[0] = 0; #7;
             SW[1] = 0;#7;
             SW[1] = 1;#7;
             SW[1] = 0;#7;
SW[0] = 1; #7;
             SW[1] = 0;#7;
             SW[1] = 1;#7;
             SW[1] = 0;#7;
             SW[1] = 1;#7;
SW[0] = 0; #7;
             SW[1] = 0;#7;
             SW[1] = 1;#7;
             SW[1] = 0;#7;
             SW[1] = 1;#7;

$stop;
// --> end
$display("Running testbench");
end
```

第二部分（同步/异步清零），清零端 SW[0]尽量多改变几次。SW[1]为输入端.

```
begin
// code that executes only once
// insert code here --> begin
CLOCK_50 = 0;SW[1] = 0;SW[0] = 0; #7;
             SW[1] = 0;#7;
             SW[1] = 1;#7;
             SW[1] = 0;#7;
SW[0] = 1; #7;
             SW[1] = 0;#7;
             SW[1] = 1;#7;
SW[0] = 0; #7;
             SW[1] = 0;#7;
             SW[1] = 1;#7;
SW[0] = 1; #7;
             SW[1] = 0;#7;
             SW[1] = 1;#7;
SW[0] = 0; #7;
             SW[1] = 0;#7;
             SW[1] = 1;#7;

$stop;
// --> end
$display("Running testbench");
end
```

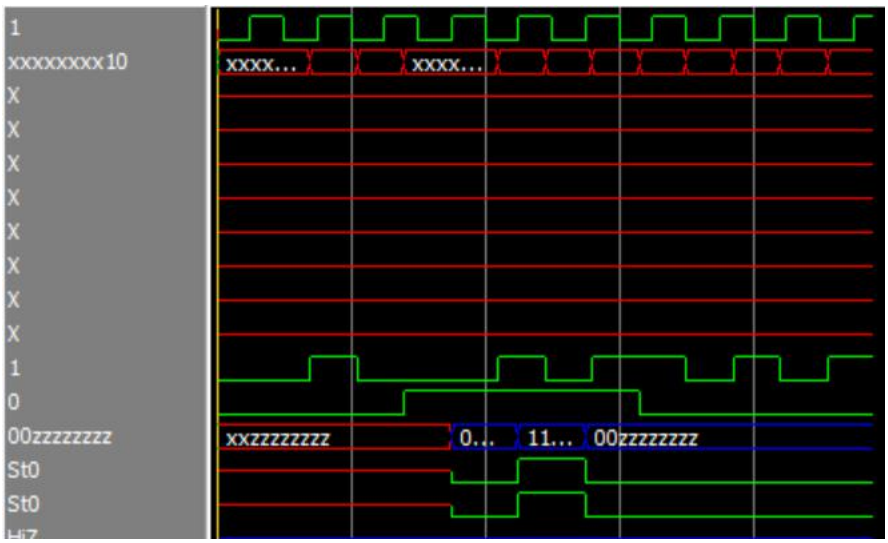
7.实验结果

仿真结果：

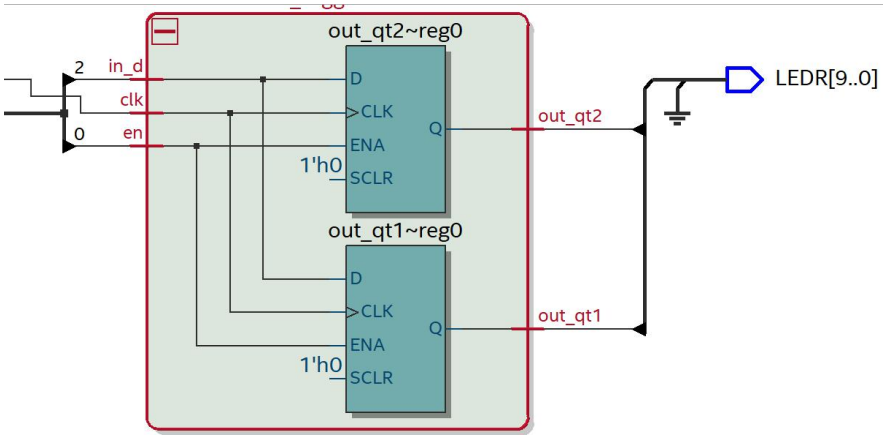
第一部分（阻塞/非阻塞）：

阻塞：

波形图：



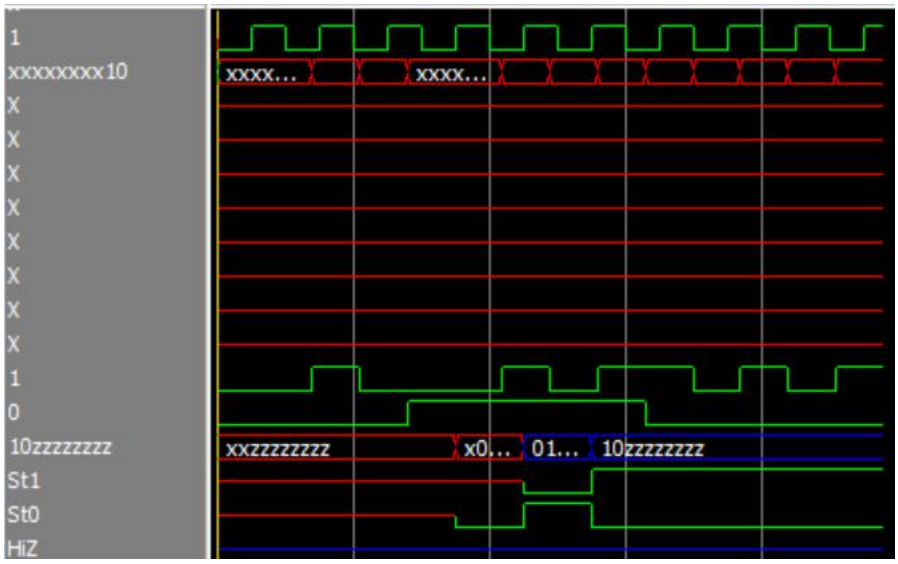
RTL viewer：



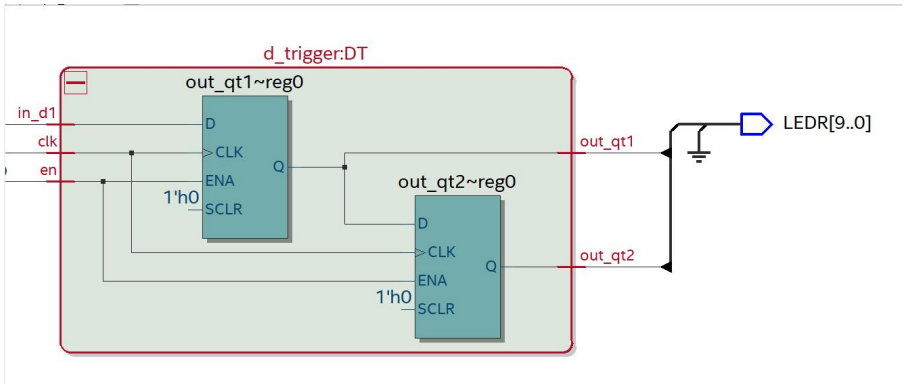
由于阻塞赋值是立即赋值，所以当 en 为 1 时，直接就将 in_d 赋值给了 out_qt1 和 out_qt2，没有级联造成的延迟。

非阻塞：

波形图：



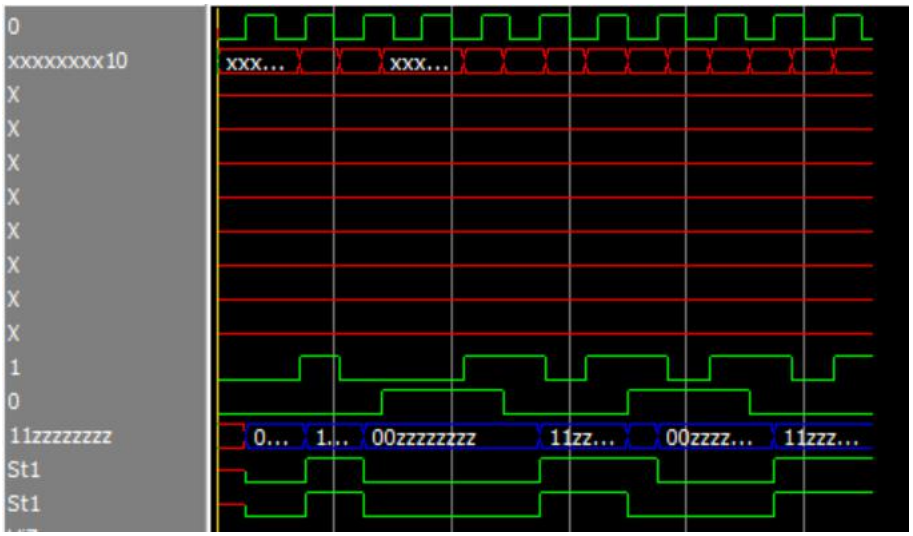
RTL viewer：



非阻塞赋值需要等 always 语句块执行完之后再进行赋值，所以就用了级联，来确保当 en 为 1 时，out_qt2 接收到的输入为 out_qt1。

第二部分（异步/同步清零）

LEDR[9]（上面的 st1）是同步清零的结果
LEDR[8]（下面的 st1）是异步清零的结果
SW[0]是清零端
SW[1]是输入端



在板子上测试时，可以用分频器赋给触发器一个周期 2s 的时钟信号，这样方便操作。

8.实验中遇到的问题

1.测试文件写的不规范，会导致测试结果不对，或者根本就出不来波形图，需要按照课件写测试文件。
2.test bench 文件一直生成不了，是因为没有在 settings 中设置 tool name 为 ModelSim-Altera。
3.波形图中的结果总是不正常，结果是没有写 always @（posedge clk）而是写了 always @(*)

9.实验得到的启示

1．学到了阻塞赋值和非阻塞赋值的不同，波形图的差异也能明显的看出：阻塞赋值语句是立即赋值语句，非阻塞赋值语句却不将结果立即赋值给表达式左边，要等到整个 always 块执行完毕后，经过一个无穷小的延时才完成赋值。
2．学到了异步清零和同步清零的不同，异步清零可以不顾时钟信号，只要清零信号到来就进行清零操作。同步清零即使清零信号有效也要等时钟信号有效沿到来时才清零。这一点从波形图的差异上也能够验证。
3．波形图不正常，在下面会有错误提示，可以根据那里的消息进行修改，有时编译找不出的问题，那里找的出来。

10.意见和建议

1. 希望之后的实验能够像这个实验一样友好。